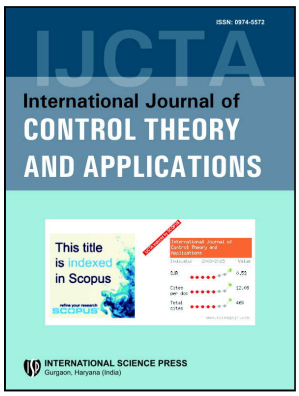

International Journal of Control Theory and Applications

ISSN : 0974-5572

© International Science Press

Volume 10 • Number 6 • 2017

Implementation of Pitch Shifter using Phase Vocoder Algorithm on Artix-7 FPGA

Nagapuri Srinivas[1] Mohith Amara[1] and Puli Kishore Kumar[1]

[1,2,3] Department of Electronics and Communication Engineering National Institute of Technology Patna, Bihar, India, Emails: ns@nitp.ac.in, mohith134896@nitp.ac.in, pulikishorek@nitp.ac.in

Abstract: The concept of pitch shifting has many advantages in different domains. Particularly in producing voice animated movies and cartoons, for enhancing the music systems, dubbing instruments it is very much needed. The task of pitch shifting is achieved using phase vocoder algorithm. It is implemented on Xilinx Artix-7 (XC7a200T-1SBG484C) FPGA. Results showed low latency and less power consumption which are suitable for real time applications

Keywords: Pitch shifting, Phase vocoder algorithm, Artix-7

1. INTRODUCTION

Now days many things has become smarter. Few of them were automatic enquiry system, cartoon voice generation, dubbing instrument in movies and few audio devices which makes music melodious. In each and every case pitch shifting is the main backbone. Pitch is nothing but the frequency with which the vocal chords vibrate. Since every person has almost a unique pitch, there exist different (unique) voices which are uncorrelated. If the pitch of a speech signal is changed then the speaker of the speech can be changed, it means this appears like the speech of some other speaker.

Pitch shifter is nothing but a device that is used for changing the pitch of the voice/speech signal. [1]. It is also a type of digital audio effect which changes the pitch of music or sound without altering the speed [5]. By scaling the frequencies with constant factor this can be done.

Many techniques in both time and frequency domain was developed and reported in the literature. Each technique was further improved over the years to increase the quality of the result and to deal with various artifacts [9], [10], [11]. Latency is one of the major issue in live systems where the voice is recorded, pitch shifted and played back in real-time. High latencies of value 50ms is acceptable for melodic instruments such as pipes or strings. But recent work has reported that latencies as low as 10ms is required for instruments with sharp attacks like drums [12]. With higher latencies, the delay prevents the player from synchronizing with the rest of the performance.

The later part of the paper is divided as follows.

Methodologies and the algorithmic part is discussed in section II followed by its analysis and implementation for real time applications in section III. Implementation results were discussed in section IV and draw the conclusion in section V.

2. METHODOLOGIES

The simplest method for shifting pitch is playing back the audio signal at different sampling rates than it was previously recorded. The duration of the audio signal will be halved if we double the play speed resulting in a pitch shift along with change in duration of the signal. But if the signal is time stretched first and resampled at higher speed then the pitch will be shifted, keeping the duration of the signal constant.

There are many more algorithms in time domain and frequency domain for doing this. In time domain, there are SOLA (synchronous overlap and add) [3], [4], PSOLA (pitch synchronous overlap and add), sinusoidal signal modeling [7], [8] are different algorithms that are generally used. In these algorithms pitch of the audio signal is found using auto-correlation and then changed. But it fails if calculated pitch was wrong. In frequency domain techniques generally phase vocoder [2], [6] and sinusoidal spectral modeling were used. Both domains have their pros and cons. Time domain techniques were used for shorter duration audio signals and particularly the audio should be from a single source. Frequency domain techniques shows best results in longer durations and audio signal consisting of multiple sources. Phase vocoder algorithm [13] is used in this paper.

In this paper, pitch has been shifted in semitones. The relation between final frequency and initial frequency with the semitone is mentioned in Eq. (1). Here semitones are denoted with 'p'

$$f_{final} = f_{initial} \times 2^{(p/12)} \tag{1}$$

Phase vocoder has three stages which are explained below.

2.1. Analysis

Since the speech signal is of larger duration windowing is done to it means splitting the signal into frames. The frames in initial signal are taken in such a way that they overlap 75% with the successive one. By changing the

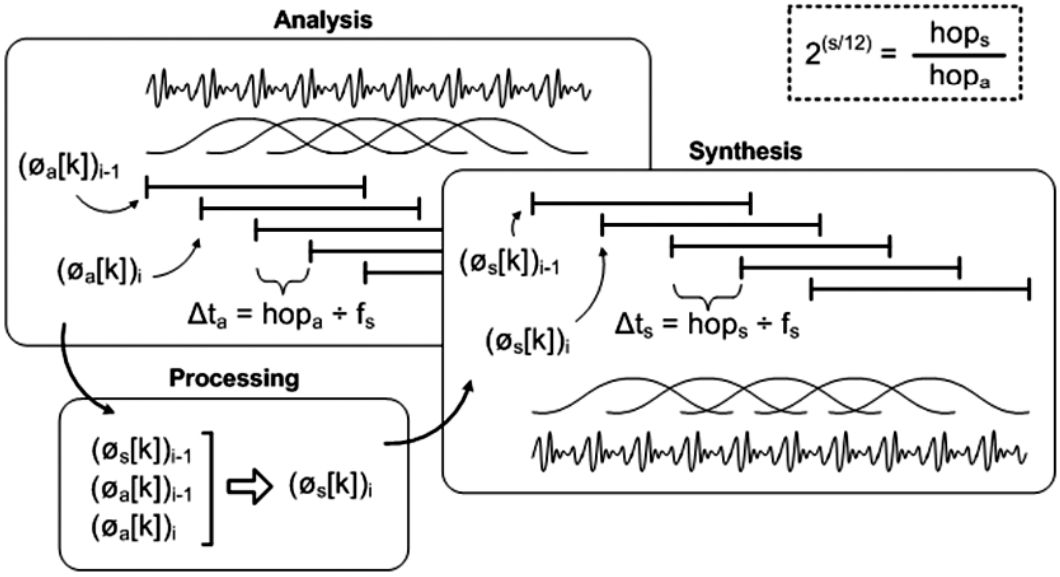

Figure 1: Overview of phase vocoder algorithm and the stages involved

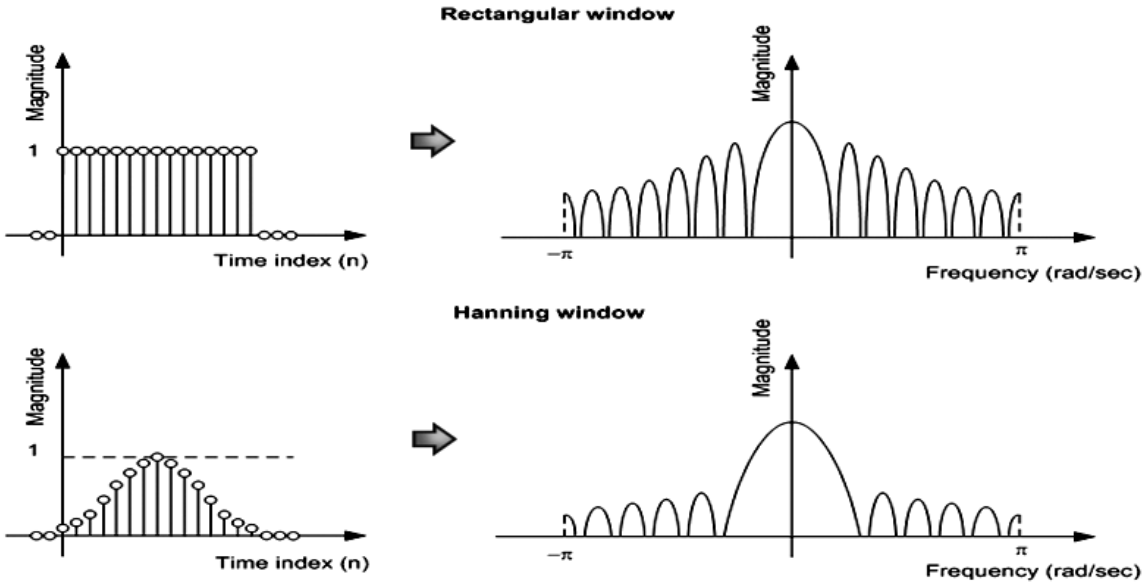

Figure 2: Spectrum of Rectangular and Hanning windows

overlap percent, we can time stretch or compress the signal. After windowing the spectrum of the signal changes due to convolution of the signal with the window. Both Hanning and rectangular windows can be used. But the energy in the side lobes of Hanning window is lesser than rectangular. And also, most of the energy is concentrated at DC component like an impulse. But due to simplicity in implementation rectangular window is used in the present work.

Now Fast Fourier Transform (FFT) is applied to the frame of N samples using the following equation.

$$(X_a[k])_i = \sum_{n=0}^{N-1} x[n + (hop_a) \times i] w[n] e^{-j\left(\frac{2\pi kn}{N}\right)} \quad k = 0,1,2\ldots,\text{N-1} \tag{2}$$

In the above equation, $x[n]$ is the speech signal, $w[n]$ is Hanning window and $(X_a[k])_i$ represents the discrete spectrum of the i^{th} frame. For better resolution of the spectrum, 75% overlap of the window is taken. The number of samples between two successive windows is referred to as the hop size (hop_a) and is equal to $\frac{N}{4}$ for an overlap of 75%.

2.2. Processing

Phase coherence problem will be occurred while performing the FFT for each frame because the frequency component in each frame of the speech signal varies. This problem is rectified using the phase information of the bins in the frames. For example, consider two sine waves with two different frame lengths as shown in figure 3.

After applying N-point FFT for these frames then N-bins will be formed ranging from 0 to $\frac{(N-1)}{N} f_s$ with an interval of $\frac{f_s}{N}$ with f_s as sampling frequency. Signal with frequency present between two bins will change the spectrum and its energy will be spread over the nearby bins. The first sine wave has a frequency of $\frac{f_s}{N}$ and falls exactly on the first bin frequency. The second wave has a frequency slightly greater than the first bin frequency.

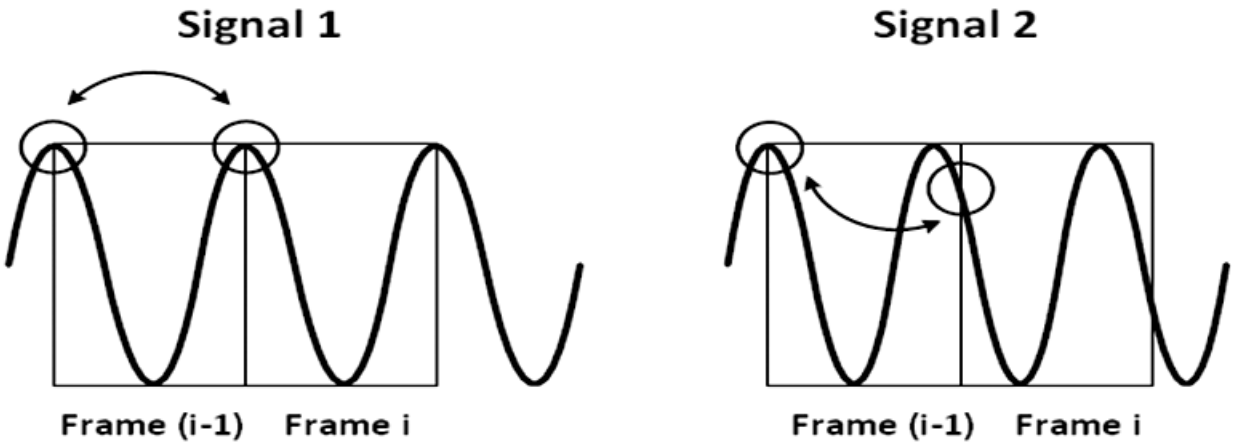

Figure 3: Sine waves with two different frame lengths

The phase difference between the two frames is referred to as the phase shift $(\Delta\phi_a[k])_i$ where k is bin index and i is frame index. Phase shift is used for finding true frequency of the bin. But the phase information calculated using FFT is wrapped, it means $(\Delta\phi_a[k])_i$ lies between $-\pi$ and π. The true frequency $(\omega_{true}[k])_i$ without wrapping can be easily obtained from the phase shift $(\Delta\phi_a[k])_i$ and the time interval Δt_a between two frames as given in Eq. (3). The time interval Δt_a is the hop size hop_a divided by the sampling rate frequency f_s.

$$(\omega_{true}[k])_i = \frac{(\Delta\phi_a[k])_i}{\Delta t_a} \tag{3}$$

First the frequency deviation from the bin is calculated and then wrapped. For obtaining the true frequency of the component this value is added to the bin frequency within the frame. The variables $(\phi_a[k])_{i-1}$ and $(\phi_a[k])_i$ stand for the phase of the previous frame and the current frame respectively. And, $\omega_{bin}[k]$ is bin frequency, $(\Delta\omega[k])_i$ means frequency deviation and $(\Delta\omega_{wrapped}[k])$ is wrapped frequency deviation. Eq. (4), Eq. (5), and Eq. (6), represent these variables.

$$(\Delta\omega[k])_i = \frac{(\phi_a[k])_i - (\phi_a[k])_{i-1}}{\Delta t_a} - \omega_{bin}[k] \tag{4}$$

$$(\Delta\omega_{wrapped}[k])_i = \text{mod}[((\Delta\omega[k])_i + \pi), 2\pi] - \pi \tag{5}$$

$$(\omega_{true}[k])_i = (\Delta\omega_{wrapped}[k]) + \omega_{bin}[k]_i \tag{6}$$

By adding phase shift new frequency of each bin is calculated. This is done by multiplying the true frequency with the time interval of the synthesis stage as givenn in Eq. (7).

$$(\phi_s[k])_i = (\phi_s[k])_{i-1} + (\omega_{true}[k])_i \times \Delta t_s \tag{7}$$

The new spectrum is calculated using Eq. (8).

$$\left|(X_s[k])_i\right| = \left|(X_a[k])_i\right| \qquad \angle(X_s[k])_i = (\phi_i)_s \tag{8}$$

2.3. Synthesis

Now the phase in frequency domain of current frame is known. Using these new values, Inverse Discrete Fourier transform (IDFT) of the frame is calculated there by converting frame into time domain. Later hanning window is applied to each frame for smoothing the signal using Eq. (9) the above mentioned are calculated.

$$q_i[n] = \left\{ \frac{1}{N} \sum_{k=0}^{N-1} (X_s[k])_i \, e^{-j\left(\frac{2\pi kn}{N}\right)} \right\} w \qquad n = 0,1,2,\ldots,N. \tag{9}$$

Here $q_i[n]$ is the time domain signal of i^{th} frame. Each frame is then overlap-added using the following eequation.

$$y[n] = \sum_{i=0}^{m-1} q_i[n-(i \times hop_s)]\{u[n-(i \times hop_s)] - u[n-(i \times hop_s)-N]\} \tag{10}$$

In the above equation m is for number of frames and $u[n]$ is unit step function.

2.4. Resampling

The obtained signal is stretched or compressed in time without any change in pitch. Now this signal is resampled to get back to the initial duration and shift the pitch.

3. ANALYSIS AND IMPLEMENTATION

We analyzed phase vocoder algorithm for pitch shifting by giving sine wave of fundamental frequency of 50Hz as input, this is shown in fig4. Using Eq. (1) and taking f_{in} as 50Hz, with p value 4 for upshift of 4 semitones, f_{out} is calculated as 64Hz, by taking p as -4 for downshift of 4 semitones gives f_{out} as 40Hz. The output results implemented using MATLAB R2013a on intel Core i3-5010U CPU with latency of 12ms are shown in fig 5 and fig 6. The same phase vocoder algorithm is synthesized using System Generator 2016.2, fig 7 and implemented using Vivado 2016.2 fig 8.

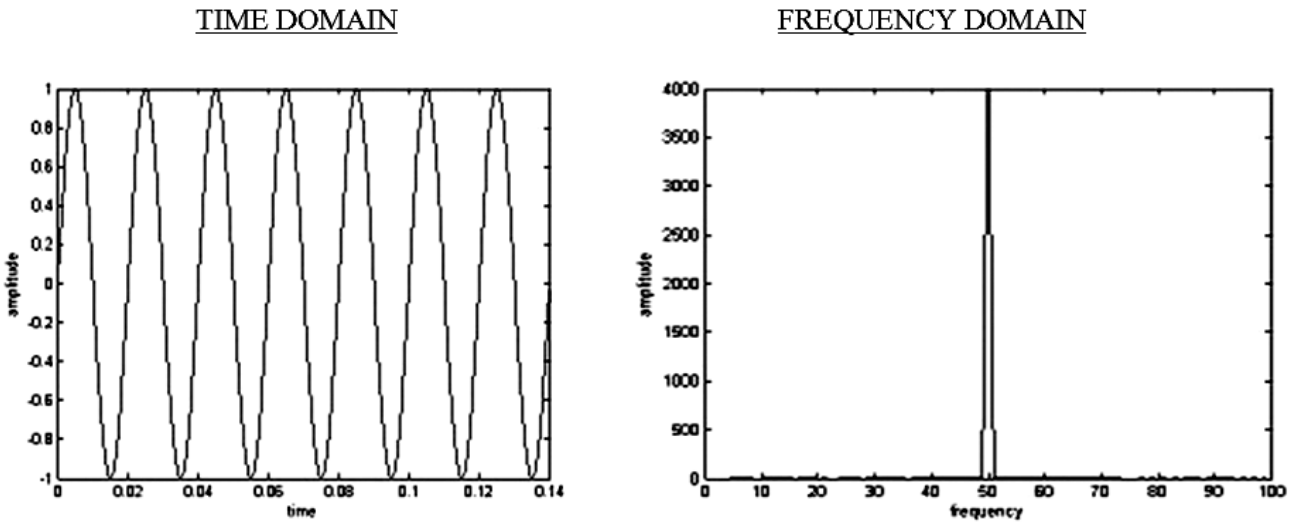

Figure 4: Input signal

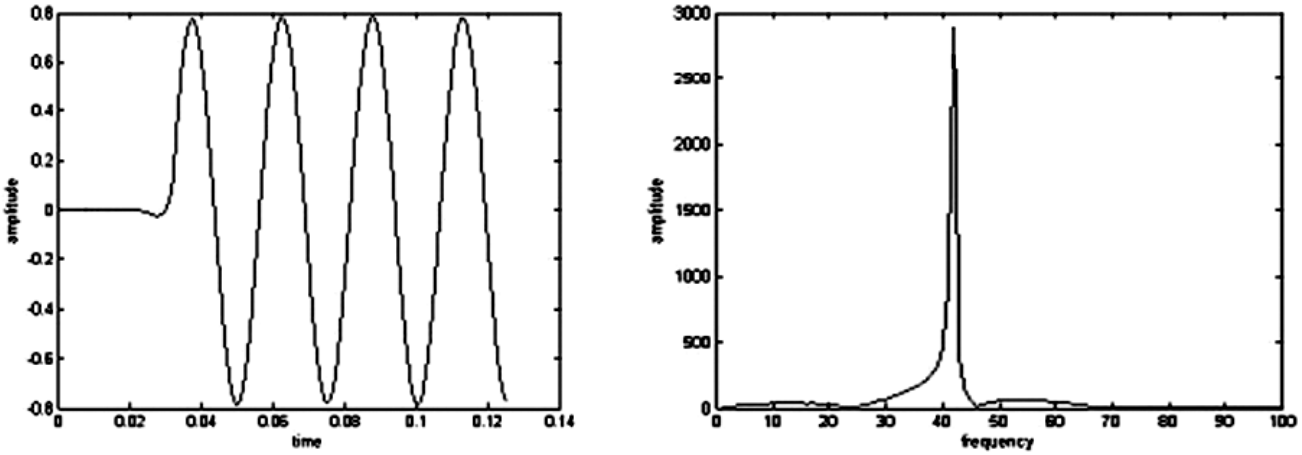

Figure 5: Output signal with 4 semitones down pitch shift

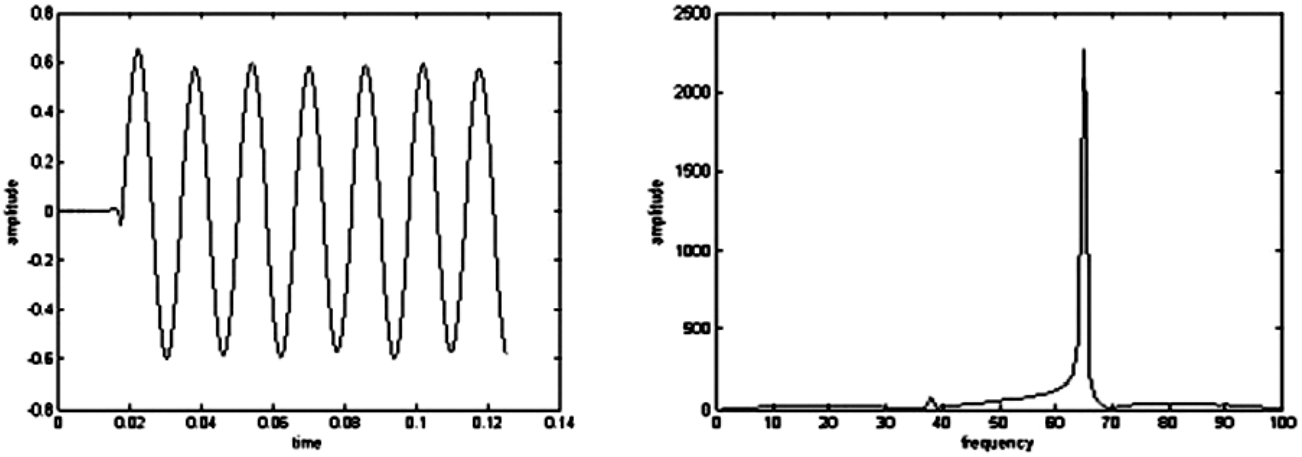

Figure 6: Output signal with 4 semitones up pitch shift

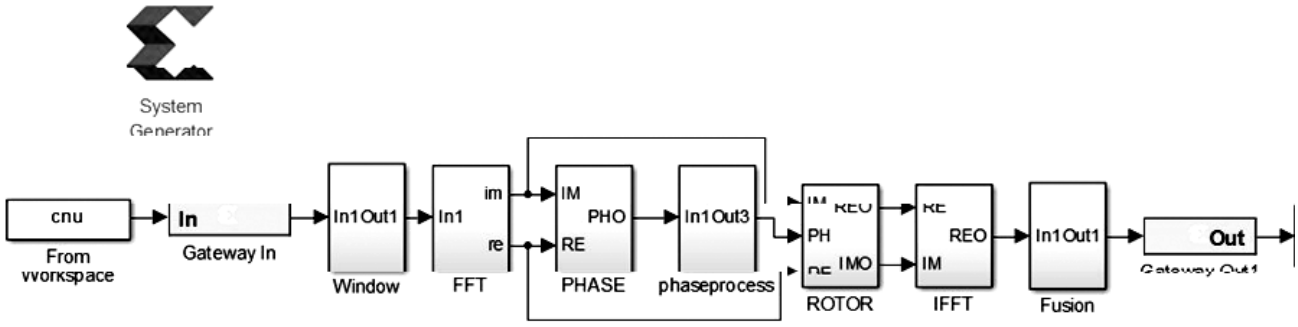

Figure 7: IP Integration using system generator

4. RESULTS

The phase vocoder algorithm is implemented on Xilinx Artix-7 FPGA (XC7A200T-1SBG484C) using System Generator 2016.2 and Vivado 2016.2 tools. The timing report, power report and hardware utilization are shown below

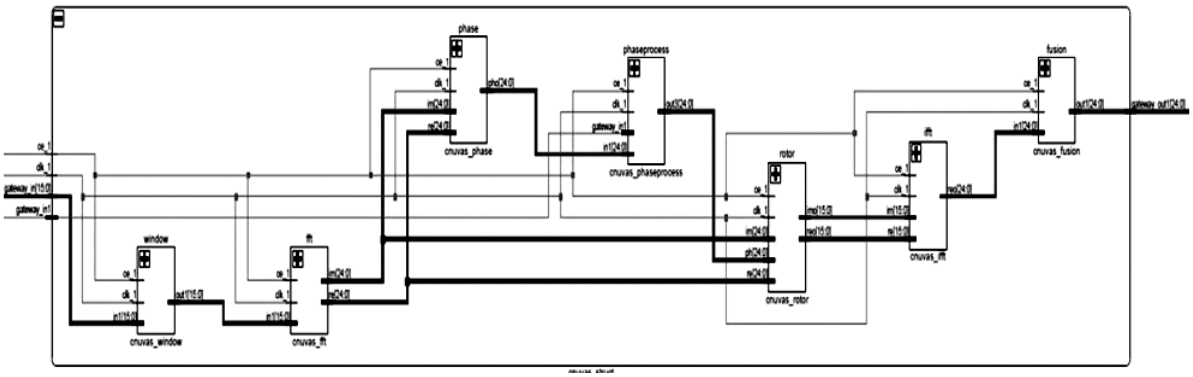

Figure 8: Schematic Diagram in Vivado

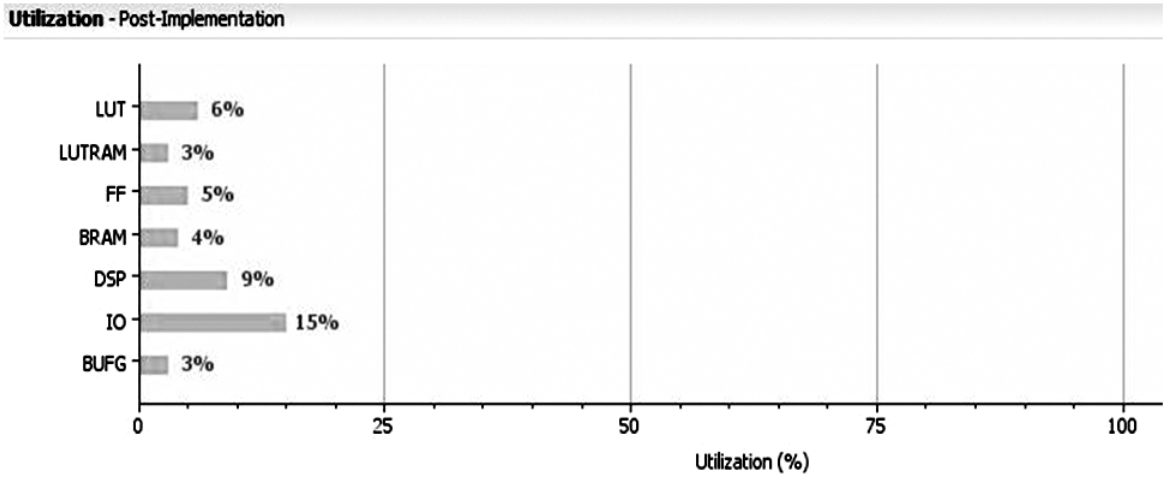

Figure 9: Utilization of Hardware

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	1.086 ns	Worst Hold Slack (WHS):	0.023 ns	Worst Pulse Width Slack (WPWS):	4.020 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	29658	Total Number of Endpoints:	29658	Total Number of Endpoints:	15947

Figure 10: Timing Summary

Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.429 W**
Junction Temperature:	**26.4 °C**
Thermal Margin:	58.6 °C (17.3 W)
Effective ϑJA:	3.3 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

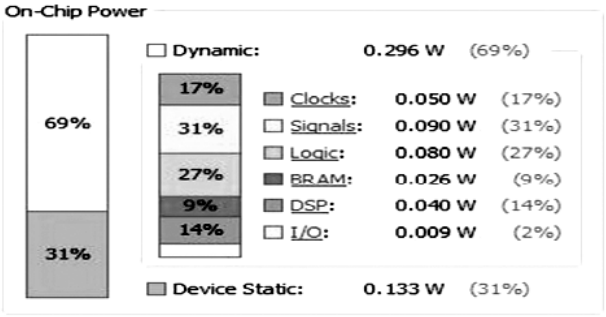

Figure 11: Power Summary

5. CONCLUSION

This paper implemented a dedicated hardware for novel low latency pitch shifter using phase vocoder algorithm. Many audio signals were validated and with required explanation of its resistance to low latency constraint is reported in the paper. In practical situations like live processing where the audio signal has to be captured, processed and played back in real-time, low latency constraint of having value less than 10ms is satisfied. The implementation on FPGA shows a latency of 8ms which is very low, so that a human ear can tolerate both the original and transformed audio signals.

REFERENCES

[1] Nicolas Juillerat and Béat Hirsbrunner, "Low latency audio pitch shifting in the frequency domain," in *IEEE International Conference on Audio Language and Image Processing (ICALIP)*, Shanghai, China, 2010.

[2] L, Flanagan J and M, Golden R, "Phase Vocoder," *The Bell System Technical Journal*, vol. 45, no. 9, pp. 1493-1509, November 1966

[3] David Dorran, "Audio time-scale modification," Dublin Institute of Technology, Dublin, PhD Thesis 2005.

[4] J Laroche, "Autocorrelation method for high-quality time/pitch-scaling," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New York, 1993.

[5] Juillerat Nicolas, Schubiger-Banz Simon, and Müller Arisona Stefan, "Low Latency Audio Pitch Shifting in the Time Domain ," in *IEEE International Conference on Audio, Language and Image Processing*, Shanghai, China, 2008, pp. 29-35.

[6] J Laroche and M Dolson, "New Phase-Vocoder Techniques for Real-Time Pitch-Shifting, Chorusing, Harmonizing and Other Exotic Audio Modifications," *Journal of the Audio Engineering Society*, vol. 47, no. 11, pp. 928-936, 1999.

[7] Tony S. Verma and Teresa H Y Meng, "Time Scale Modification Using a Sines+Transients+Noise Signal Model," in *Digital Audio Effects Workshop*, Barcelona, 1998, pp. 49-52.

[8] S. N. Levine, J. O. Smith, A Sines+Transients+Noise Audio Representation for Data Compression and Time/Pitch Scale Modifications, 105th Audio Engineering Society Convention, San Francisco, 1998.

[9] David Dorran, Euqene Coyle, and Robert Lawlor, "An Efficient Phasiness Reduction Technique for Moderate Audio Time-scale Modification," in *In International Conference on Digital Audio Effects*, Naples, Italy, 2004.

[10] S.N. Levine, "Multiresolution sinusoidal modeling for wideband audio with modifications," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Seattle, 1998, pp. 3585-3588.

[11] J Laroche and M Dolson, "Phase-vocoder: about this phasiness business," in *IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics*, New York, 1997.

[12] . N. Lago, "The Quest for Low Latency" *Proc. of the International Computer Music Conference*, Miami, Florida, pp. 33 – 36, 2004.

[13] "Pitch shifter algorithm," (Date last accessed 16-Nov-2016). [Online]. Available: http://www.guitarpitchshifter.com/algorithm.html.